IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application. No:	09/976,726
Filed:	October 12, 2001
Inventor(s):	Jason King, Craig Smith, Greg McKaskle and Andrew Dove
Title:	SYSTEM AND METHOD FOR ENABLING A GRAPHICAL PROGRAM TO RESPOND TO USER INTERFACE EVENTS
Examiner:	Zhou, Ting G.
Group/Art Unit:	2173

§

Atty. Dkt. No: 5150-58700

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed toCommissioner for Patents, Alexandria, VA 22313-1450, on the date indicated below.

Mark S. Williams

6-12-06 (Date) Mark S. Williams (Signature)

APPEAL BRIEF

Box: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

Further to the Notice of Appeal filed April 11, 2006, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

06/16/2006 MAHMED1 00000024 501505 09976726
01 FC:1402 500.00 DA

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expressway, Bldg. B, Austin, Texas 78759-3504.

II. RELATED APPEALS AND INTERFERENCES

No related appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-25, 32-44, 46-60, and 62-89 are pending in the application. All of the pending claims stand rejected and are the subject of this appeal. A copy of the claims as incorporating entered amendments is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

All amendments have been entered, including the most recent amendment filed on April 11, 2006 with the Notice of Appeal. The Claims Appendix hereto reflects the current state of the claims.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Independent claim 1 relates to a method for configuring a graphical program to respond to one or more user interface events. As described in more detail in the Description of the Related Art of the present application, a user typically creates a graphical program by arranging nodes or icons in a block diagram and interconnecting them such that the block diagram graphically displays a procedure or method for accomplishing a certain result.

The method of claim 1 comprises creating a graphical user interface for the graphical program in response to first user input. For example, the user may include various types of user interface controls or indicators in the graphical user interface, such as charts, graphs, push buttons, knobs, numeric controls, text boxes, list boxes, check boxes, etc. (*See block 202 of Figure 4 and p. 19, lines 6-23*)

The method further comprises displaying a first node for receiving user interface events in a block diagram for the graphical program in response to second user input. For example, the user may drag and drop the first node into the block diagram from a palette, may select the first node using a menu command, or may cause the first node to be displayed in the block diagram in any of various other ways. (*See block 302 of Figure 5; p. 24, lines 22-28*).

The method further comprises receiving third user input explicitly specifying one or more user interface events to configure for the first node. For example, in one embodiment the user may specify the desired user interface events to configure for the first node by interacting with a graphical user interface dialog or wizard. In another embodiment, the user may specify the one or more user interface events to configure for the first node by connecting other nodes to the first node, where the other nodes specify the one or more user interface events. (*See block 304 of Figure 5; p. 22, line 4 – p. 23, line 2; p. 25, lines 1-4*).

The method further comprises configuring the first node to receive the one or more user interface events explicitly specified by the third user input during execution of the graphical program. Configuring the first node to receive the one or more user interface events comprises configuring the first node to receive notification when the one

or more user interface events are generated during execution of the graphical program. (*See block 206 of Figure 4; p. 20, lines 19-26*).

The method further comprises associating one or more portions of graphical code with the first node in response to fourth user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the first node is configured to receive. Each portion of graphical code may comprise a plurality of interconnected nodes that visually indicate functionality for responding to one or more of the user interface events which the first node is configured to receive. Thus, when one of the user interface events which the first node is configured to receive is generated during execution of the graphical program, a corresponding portion of graphical code may execute to respond to the user interface event. (*See Figures 7, 13, and 14; See also p. 21, line 8 – p. 22, line 3; p 23, lines 11-18; p. 25, lines 5-10*).

Claim 32 is a memory medium claim analogous to the method claim 1 and recites similar limitations as claim 1.

The method of claim 19 recites similar subject matter as claim 1. The method comprises displaying a first node for receiving programmatic events in a block diagram for the graphical program in response to first user input. (*See block 302 of Figure 5; p. 24, lines 22-28*).

The method further comprises receiving second user input explicitly specifying one or more programmatic events to configure for the first node. (*See block 304 of Figure 5; p. 22, line 4 – p. 23, line 2; p. 25, lines 1-4*).

The method further comprises configuring the first node to receive the one or more programmatic events explicitly specified by the second user input during execution of the graphical program. (*See block 206 of Figure 4; p. 20, lines 19-26*).

The method further comprises associating one or more portions of graphical code with the first node in response to third user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the programmatic events

which the first node is configured to receive. (*See Figures 7, 13, and 14; See also p. 21, line 8 – p. 22, line 3; p 23, lines 11-18; p. 25, lines 5-10*).

The method of claim 23 comprises creating a graphical user interface for the graphical program in response to user input. (*See block 202 of Figure 4 and p. 19, lines 6-23*).

The method further comprises creating a block diagram for the graphical program in response to user input, where creating the block diagram comprises creating a first portion of graphical code in response to user input, wherein the first portion of graphical code includes one or more nodes for responding to user interface events. (*See blocks 204 and 206 of Figure 4; p. 21, lines 8-17; See Figures 7, 13, and 14*).

The method further comprises receiving user input explicitly specifying one or more user interface events to associate with the first portion of graphical code. (*See p. 22, line 4 – p. 23, line 2*).

The method further comprises configuring the first portion of graphical code to execute in response to the one or more explicitly specified user interface events being generated during execution of the graphical program. (*See p. 21, line 8 – p. 22, line 3; p 23, lines 11-18; p. 25, lines 5-10; See Figures 7, 13, and 14;*).

The method of claim 36 comprises displaying a first node in a block diagram of the graphical program in response to user input. (*See block 302 of Figure 5; p. 24, lines 22-28*).

The method further comprises associating a first portion of graphical source code with the first node in response to user input, wherein the first portion of graphical source code associated with the first node comprises a plurality of interconnected nodes. (*See p. 21, line 8 – p. 22, line 3; p 23, lines 11-18; p. 25, lines 5-10; See Figures 7, 13, and 14*).

The method further comprises associating a first user interface event with the first node in response to user input explicitly specifying the first user interface event. (*See block 304 of Figure 5; p. 22, line 4 – p. 23, line 2; p. 25, lines 1-4*).

The method further comprises configuring the first portion of graphical source code associated with the first node to execute in response to the first user interface event

associated with the first node being generated during execution of the graphical program. (*See Figures 7, 13, and 14; See p. 21, line 8 – p. 22, line 3; p 23, lines 11-18; p. 25, lines 5-10*).

Claim 53 is a memory medium claim analogous to the method claim 36 and recites similar limitations as claim 36.

Claim 66 is a system claim analogous to the method claim 36 and recites similar limitations as claim 36.

The method of claim 86 comprises displaying a first node for receiving programmatic events in a block diagram of the graphical program in response to user input. (*See block 302 of Figure 5; p. 24, lines 22-28*).

The method further comprises associating a plurality of portions of graphical code with the first node in response to user input, wherein each portion of graphical code comprises one or more nodes. (*See p. 21, line 8 – p. 22, line 3; p 23, lines 11-18; p. 25, lines 5-10; See Figures 7, 13, and 14*).

The method further comprises receiving user input explicitly specifying one or more programmatic events to which each of the portions of graphical code associated with the first node corresponds. (*See block 304 of Figure 5; p. 22, line 4 – p. 23, line 2; p. 25, lines 1-10*).

The method further comprises configuring each portion of graphical code associated with the first node to execute in response to the one or more programmatic events to which the portion of graphical code corresponds being generated during execution of the graphical program. (*See p. 21, line 8 – p. 22, line 3; p 23, lines 11-18; p. 25, lines 5-10; See Figures 7, 13, and 14*).

The method of claim 87 comprises creating a first portion of graphical code in response to user input, wherein the first portion of graphical code comprises one or more nodes that visually indicate functionality for responding to programmatic events generated during execution of the graphical program. (*See p. 21, lines 8-17; See Figures 7, 13, and 14*).

The method further comprises configuring the graphical program to dynamically register a first programmatic event during execution of the graphical program. Dynamically registering the first programmatic event comprises dynamically associating the first programmatic event with the first portion of graphical code. Dynamically registering the first programmatic event causes the first portion of graphical code to execute in response to the first programmatic event being generated. For example, it may only be necessary or desirable to receive and respond to the first programmatic event if some condition becomes true during the course of executing the graphical program. Thus, the first programmatic event may be dynamically registered when the condition becomes true or at some pre-determined point in the graphical program. (*See p. 22, line 19 – p. 23, line 10*).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-7, 10-25, 32-44, 46-60, 62-79, and 82-89 stand rejected under 35 U.S.C. 102(b) as being anticipated by Cain et al., U.S. Patent No. 5,651,108 (hereinafter "Cain").

Claims 8-9 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Cain in view of Zizzo, U.S. Patent No. 6,578,174 (hereinafter "Zizzo").

Claims 16, 17, 80-81, and 87 stand rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement.

VII. ARGUMENT

Section 102(b) Rejection

Claims 1-7, 10-25, 32-44, 46-60, 62-79, and 82-89 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Cain. Appellant respectfully traverses this rejection for the following reasons. Different groups of claims are addressed under respective subheadings.

Claims 1, 19, 32

Cain relates generally to a relational database management system (RDBMS) in which the user places screen objects (e.g, boxes, screen buttons, table objects, and the like) on an on-screen window or "form". The screen objects have associated properties, as well as methods that execute in response to events.

Appellant respectfully submits that Cain does not teach the subject matter recited in the present claims. For example, claim 1 recites as follows:

> 1. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:
> creating a graphical user interface for the graphical program in response to first user input;
> displaying a first node for receiving user interface events in a block diagram for the graphical program in response to second user input;
> receiving third user input explicitly specifying one or more user interface events to configure for the first node;
> configuring the first node to receive the one or more user interface events explicitly specified by the third user input during execution of the graphical program; and
> associating one or more portions of graphical code with the first node in response to fourth user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the first node is configured to receive.

In the rejection of claim 1, the Examiner refers to the section at Col. 10, line 48 – Col. 13, line 17, in which Cain describes an example of building a program which provides a customized button that responds to a mouse click event by displaying a dialog box. The button includes built-in methods that execute in response to events (Col. 3,

lines 45-55 and Col. 11, lines 55-59). In particular, the button includes a built-in "pushButton" method that executes in response to push-button events which are generated when the user clicks the button during execution of the program. Cain teaches that the user customizes the pushButton method to make it respond to a push-button event by displaying a dialog box (Col. 12, lines 16-18).

As per the limitation of, "receiving third user input explicitly specifying one or more user interface events to configure for the first node," the Examiner refers to the description of FIGS. 4D-4E. The Examiner asserts that, "users can change or attach new user interface events which the button will respond to via input on the graphical user interface". Appellant respectfully disagrees.

FIG. 4D illustrates a popup menu from which the programmer can select a button property, such as Button Type, Center Label, or Design, or select the Methods menu item (Col. 11, lines 30-44). Selecting the Methods menu item causes the Methods window shown in FIG. 4E to be displayed. The Methods window includes a Built-in Methods listbox which lists the built-in (default) methods already attached to the button (Col. 11, lines 62-67) and a Custom Methods box which lists the names of additional custom (user-defined) methods that have been attached to the button (Col. 12, lines 7-9). Cain teaches that, "The behavior of the button object can be varied in two ways: the built-in methods can be edited or new custom methods can be written" (Col. 12, lines 14-16). Thus, the graphical user interface in FIG. 4E allows the user to change the built-in methods attached to the button or attach new custom methods to the button, but Cain teaches nothing about receiving user input explicitly specifying one or more user interface events to configure for the button.

Thus, Appellant respectfully submits that the Examiner has confused a method with an event. Cain clearly teaches that a method is not the same as an event. (See for example Col. 11, lines 55-56: "Every object in a form (as well as the form itself) includes built-in methods that execute in response to events." *Emphasis added.*)

Claim 1 recites, "receiving third user input explicitly specifying one or more user interface events to configure for the first node" and "configuring the first node to receive the one or more user interface events explicitly specified by the third user input during execution of the graphical program". Cain clearly teaches that certain events, such as the

push-button event, are attached to the button by default. Cain does not teach configuring the button to receive one or more user interface events that have been explicitly specified by the user, as recited in claim 1.

Furthermore, Cain does not teach, "associating one or more portions of graphical code with the first node in response to fourth user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the first node is configured to receive." On the contrary, Cain teaches that the methods which respond to the button events comprise text-based code, i.e., program code constructed in a text-based programming language, as opposed to graphical code which comprises one or more nodes. For example, as shown in FIGS. 4F – 4H, the pushButton method which executes to respond to the push-button event is constructed from text-based code, not graphical code.

For at least the reasons given above, Appellant respectfully submits that Cain does not teach numerous limitations recited in claim 1, and thus, claim 1 is patentably distinct over Cain. Inasmuch as claims 19 and 32 recite similar limitations as claim 1, Appellant respectfully submits that claims 19 and 32 are also patentably distinct over Cain.

Claims 2, 20, 33

Claims 2, 20, and 33 recite additional limitations not taught by Cain. For example, claim 2 recites the additional limitations of, "wherein the first node comprises one or more sub-diagrams, wherein said associating the one or more portions of graphical code with the first node comprises displaying each portion of graphical code within one of the sub-diagrams of the first node."

In the rejection of claim 2, the Examiner cites a portion of Cain pertaining to the notion of containership, whereby objects may be contained within other objects. However, Cain does not teach the specific limitations recited in claim 2. As discussed above with reference to claim 1, the Examiner has equated the "first node" recited in claims 1 and 2 with Cain's button. However, Cain's button does not comprise one or more sub-diagrams, and Cain does not teach displaying portions of graphical code within sub-diagrams of the button.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claims 3, 21, 34

Claims 3, 21, and 34 recite additional limitations not taught by Cain. For example, claim 3 recites the additional limitations of:

> "wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying one or more user interface events to which each of the sub-diagrams of the first node corresponds;
>
> wherein for each portion of graphical code, displaying the portion of graphical code within one of the sub-diagrams of the first node comprises configuring the portion of graphical code to execute in response to the one or more user interface events to which the sub-diagram corresponds."

As noted above with reference to claim 2, Cain does not teach a first node which comprises one or more sub-diagrams, wherein portions of graphical code are displayed within the sub-diagrams. Also, as discussed with reference to claim 1, Cain does not teach receiving third user input explicitly specifying one or more user interface events to configure for the first node. Cain also does not teach, "receiving user input explicitly specifying one or more user interface events to which each of the sub-diagrams of the first node corresponds," as recited in claim 3.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claims 4, 35

Claims 4 and 35 recite additional limitations not taught by Cain. For example, claim 4 recites the additional limitation of, "wherein for each portion of graphical code, displaying the portion of graphical code within one of the sub-diagrams of the first node comprises displaying the one or more nodes of the portion of graphical code within one of the sub-diagrams of the first node."

As discussed above with reference to claim 1, each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the first node is configured to receive. Thus, for each portion of graphical code,

the one or more nodes of the portion of graphical code are displayed within one of the sub-diagrams of the first node. As discussed above, Cain does not teach the concept of a portion of graphical code responding to a user interface event, but instead teaches that methods constructed from text-based code respond to user interface events. Thus, Cain does not teach the concept of, "displaying the one or more nodes of the portion of graphical code within one of the sub-diagrams of the first node."

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claims 6, 7

Claims 6 and 7 recite additional limitations not taught by Cain. For example, claim 6 recites the additional limitation of, "wherein the method further comprises executing one of the portions of graphical code associated with the first node in response to each of the one or more user interface events which the first node is configured to receive being generated during execution of the graphical program." As discussed above with reference to claim 1, Cain teaches that methods execute in response to events and teaches that the methods are constructed as text-based code, not graphical code. Cain does not teach the concept of executing a portion of graphical code in response to a user interface event.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claim 12

Claim 12 recites the additional limitation of, "wherein each of the portions of graphical code associated with the first node is displayed within the first node." As discussed above, Cain does not teach portions of graphical code which are associated with the first node, where each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the first node is configured to receive. Furthermore, Cain does not teach that each of the portions of graphical code associated with the first node is displayed within the first node. The

Examiner has equated the "first node" with a push-button. Cain simply does not teach displaying portions of graphical code within a push-button.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claims 13, 14

Claims 13 and 14 recite additional limitations not taught by Cain. For example, claim 13 recites the additional limitations of:

> "wherein said configuring the first node to receive the one or more user interface events comprises configuring the first node to receive a first user interface event;
>
> wherein the first user interface event explicitly specifies a first user interface element of the graphical user interface and an action performed on the first user interface element."

As discussed above with respect to claim 1, Cain does not teach the concept of configuring a first node to receive a first user interface event. Furthermore, Cain does not teach that a first user interface event which explicitly specifies both a first user interface element of the graphical user interface and an action performed on the first user interface element. The Examiner has equated the "first node" with Cain's push-button, which is itself a user interface element. Cain teaches that the push-button includes a built-in "pushButton" method that executes in response to push-button events which are generated when the user clicks the button during execution of the program. Appellant respectfully submits that the push-button events explicitly specify the action performed on the push-button, i.e., the button click, but do not explicitly specify the push-button itself (i.e., the user interface element on which the click was performed).

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claim 15

Claim 15 recites the additional limitation of:

> "displaying a first graphical user interface dialog for configuring the first node;

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input to the first graphical user interface dialog to explicitly specify the one or more user interface events."

As discussed above with reference to claim 1, the graphical user interface of FIG. 4E allows the user to change the built-in methods attached to the button or attach new custom methods to the button, but Cain teaches nothing about the use of a graphical user interface to explicitly specify one or more user interface events to configure for the first node.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claims 16, 71

Claims 16 and 71 recite additional limitations not taught by Cain. For example, claim 16 recites the additional limitations of:

"displaying a second node for dynamically registering user interface events in the block diagram in response to user input;

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying a first user interface event to dynamically register during execution of the graphical program;

wherein the method further comprises configuring the second node to dynamically register the first user interface event during execution of the graphical program such that, after registering the event, the first node becomes operable to receive the first user interface event."

Cain does not teach dynamically registering a first user interface event during execution of the graphical program. In the rejection of claim 16, the Examiner refers to the user interacting with the graphical user interface of FIG. 4E. As discussed above, the user interacts with this graphical user interface to change the built-in methods attached to the button or attach new custom methods to the button. The user interacts with the graphical user interface while editing the graphical program, not during execution of the graphical program. Cain teaches nothing at all about an event being dynamically registered during execution of the program.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claim 17

Claim 17 recites the additional limitations of, "wherein said configuring the second node to dynamically register the first user interface event during execution of the graphical program comprises connecting the second node to the first node in response to user input." As discussed above with respect to claim 16, Cain does not teach configuring a second node to dynamically register a first user interface event during execution of the graphical program. Cain also does not teach the additional limitation that the second node is configured to dynamically register the first user interface event during execution of the graphical program by connecting the second node to the first node in response to user input.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 18

Claim 18 recites the additional limitations of:

> wherein the one or more user interface events specified by the third user input includes a first user interface event;
>
> wherein the method further comprises displaying a second node for dynamically un-registering user interface events in the block diagram in response to user input;
>
> configuring the second node to dynamically un-register the first user interface event during execution of the graphical program;
>
> wherein, before said dynamically un-registering the first user interface event, the first node is operable to receive the first user interface event;
>
> wherein, after said dynamically un-registering the first user interface event, the first node is not operable to receive the first user interface event.

Cain does not teach dynamically un-registering a first user interface event during execution of the graphical program. In the rejection of claim 18, the Examiner refers to the user interacting with the graphical user interface of FIG. 4E. As discussed above, the user interacts with this graphical user interface to change the built-in methods attached to the button or attach new custom methods to the button. The user interacts with the

graphical user interface while editing the graphical program, not during execution of the graphical program. Cain teaches nothing at all about an event being dynamically un-registered during execution of the program.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 69

Claim 69 recites the additional limitations of:

> wherein said associating the one or more portions of graphical code with the first node comprises associating a first portion of graphical code and a second portion of graphical code with the first node;
>
> wherein the first node comprises a plurality of sub-diagrams;
>
> wherein associating the first portion of graphical code with the first node comprises displaying the first portion of graphical code within a first sub-diagram of the first node; and
>
> wherein associating the second portion of graphical code with the first node comprises displaying the second portion of graphical code within a second sub-diagram of the first node.

The Examiner has equated the "first node" with the button taught by Cain. However, Cain does not teach that the button comprises a plurality of sub-diagrams and does not teach that a first portion of graphical code is associated with the button by displaying the first portion of graphical code within a first sub-diagram of the button. Cain also does not teach that a second portion of graphical code is associated with the button by displaying the second portion of graphical code within a second sub-diagram of the button. In fact, as discussed above, Cain does not teach associating portions of graphical code with the button at all, but instead teaches associating various methods with the button, where the methods are composed of text-based code, not graphical code.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 70

Claim 70 recites the additional limitation of, "displaying one or more of the portions of graphical code associated with the first node in the block diagram of the

graphical program." The Examiner has equated the "first node" with the button taught by Cain. As discussed above, Cain does not teach associating one or more portions of graphical code with the button in response to user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the button is configured to receive. Cain also does not teach displaying one or more of the portions of graphical code associated with the button in the block diagram of the graphical program.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 72

Claim 72 recites the additional limitation of, "wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying one or more user interface events to which each of the portions of graphical code associated with the first node corresponds." As discussed above with respect to claim 1, Cain does not teach receiving third user input explicitly specifying the one or more user interface events to configure for the first node. Cain also does not teach receiving user input explicitly specifying one or more user interface events to which each of the portions of graphical code associated with the first node corresponds.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 73

Claim 73 recites the additional limitation of, "wherein each of the portions of graphical code is comprised within the first node." The Examiner has equated the "first node" with the button taught by Cain. As discussed above, Cain does not teach associating one or more portions of graphical code with the button in response to user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the button is configured to receive. Cain also does not teach that the portions of graphical code are comprised within

the button. Instead, Cain teaches that various methods comprising text-based source code are associated with the button. Cain does not teach that these methods are comprised within the button.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claims 76, 77

Claims 76 and 77 recite additional limitations not taught by Cain. For example, claim 76 recites the additional limitations of:

> wherein said receiving third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying a first user interface event and a second user interface event;
>
> wherein said associating the one or more portions of graphical code with the first node comprises associating a first portion of graphical code with the first node, wherein the first portion of graphical code comprises a plurality of interconected nodes visually indicating functionality for responding to the first user interface event;
>
> wherein said associating the one or more portions of graphical code with the first node further comprises associating a second portion of graphical code with the first node, wherein the second portion of graphical code comprises a plurality of interconected nodes visually indicating functionality for responding to the second user interface event.

The Examiner has equated the "first node" with the button taught by Cain. Cain does not teach that a first portion of graphical code is associated with the button, wherein the first portion of graphical code comprises a plurality of interconnected nodes visually indicating functionality for responding to a first user interface event. Similarly, Cain does not teach that a second portion of graphical code is associated with the button, wherein the second portion of graphical code comprises a plurality of interconnected nodes visually indicating functionality for responding to a second user interface event.

Cain instead teaches that methods comprising text-based code, not graphical code, execute in response to events. The functionality for responding to the events is defined by the text-based code of the methods. The text-based code simply comprises lines of text. The text-based code does not comprise a plurality of interconnected nodes that visually indicate the event response functionality.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claim 79

Claim 79 recites the additional limitations of, "wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises graphically connecting a plurality of objects to the first node in response to user input, wherein each object directly represents a user interface event."

The Examiner has equated the "first node" with the button taught by Cain. As discussed above with respect to claim 1, Cain does not teach receiving user input explicitly specifying one or more user interface events to configure for the button. Cain also does not teach that the one or more user interface events are explicitly specified by graphically connecting a plurality of objects to the button in response to user input.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 82

Claim 82 recites the additional limitations of, "creating the portions of graphical code in response to user input, wherein for each portion of graphical code, creating the portion of graphical code comprises including one or more nodes in the portion of graphical code in response to user input, wherein the one or more nodes are operable to respond to one or more of the user interface events which the first node is configured to receive." As discussed above, Cain teaches that methods comprising text-based code are attached to the button, where the text-based code is operable to respond to user interface events. Cain does not teach creating a portion of graphical code which includes one or more nodes, wherein the one or more nodes are operable to respond to one or more of the user interface events which the first node is configured to receive.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 23

Claim 23 recites as follows:

> 23. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:
>
> creating a graphical user interface for the graphical program in response to user input;
>
> creating a block diagram for the graphical program in response to user input, wherein creating the block diagram comprises creating a first portion of graphical code in response to user input, wherein the first portion of graphical code includes one or more nodes for responding to user interface events;
>
> receiving user input explicitly specifying one or more user interface events to associate with the first portion of graphical code; and
>
> configuring the first portion of graphical code to execute in response to the one or more explicitly specified user interface events being generated during execution of the graphical program.

Cain does not teach, "receiving user input explicitly specifying one or more user interface events to associate with the first portion of graphical code." As discussed above with respect to claim 1, Cain teaches that objects such as a button are configured to receive certain events, such as the push-button event, by default. Cain does not teach configuring a portion of graphical code to receive one or more user interface events that have been explicitly specified by the user, as recited in claim 23.

Furthermore, Cain does not teach the concept of configuring a portion of graphical code to execute in response to the one or more explicitly specified user interface events being generated during execution of the graphical program. On the contrary, Cain teaches that the methods which respond to the events comprise text-based code, i.e., program code constructed in a text-based programming language, as opposed to graphical code which comprises one or more nodes. For example, as shown in FIGS. 4F – 4H, the pushButton method which executes to respond to the push-button event is constructed from text-based code, not graphical code.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 24

Claim 24 recites the additional limitations of:

> "wherein the first portion of graphical code includes a plurality of nodes, wherein said creating the first portion of graphical code in response to user input

comprises arranging the plurality of nodes on a display and interconnecting the plurality of nodes in response to user input, wherein the plurality of interconnected nodes visually indicates functionalty for responding to the one or more user interface events associated with the first portion of graphical code."

Cain does not teach the concept of creating a first portion of graphical code which includes a plurality of interconnected nodes, where the plurality of interconnected nodes visually indicates functionalty for responding to the one or more user interface events associated with the first portion of graphical code. Cain instead teaches that methods comprising text-based code execute in response to events. The functionality for responding to the events is defined by the text-based code of the methods. The text-based code simply comprises lines of text. The text-based code dos not comprise a plurality of interconnected nodes that visually indicate the event response functionality.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claim 25

Claim 25 recites the additional limitations of:

executing the graphical program;

wherein one or more user interface events associated with the first portion of graphical code are generated during execution of the graphical program;

wherein the method further comprises executing the first portion of graphical code in response to the one or more user interface events associated with the first portion of graphical code being generated.

As discussed above, Cain teaches that methods execute in response to events and teaches that the methods are constructed as text-based code, not graphical code. Cain does not teach the concept of executing a portion of graphical code in response to a user interface event.

Appellant thus submits that Cain does not teach the limitations recited in this claim.

Claims 36, 53, 66, 86

Claim 36 recites as follows:

36. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:

displaying a first node in a block diagram of the graphical program in response to user input;

associating a first portion of graphical source code with the first node in response to user input, wherein the first portion of graphical source code associated with the first node comprises a plurality of interconnected nodes;

associating a first user interface event with the first node in response to user input explicitly specifying the first user interface event; and

configuring the first portion of graphical source code associated with the first node to execute in response to the first user interface event associated with the first node being generated during execution of the graphical program.

As discussed above with reference to claim 1, the Examiner has equated the "first node" with a button such as taught by Cain. With respect to the limitation of, "associating a first portion of graphical source code with the first node in response to user input, wherein the first portion of graphical source code associated with the first node comprises a plurality of interconnected nodes", the Examiner refers to Col. 10, lines 50-58, where Cain teaches that "the example shows how to attach program code (Borland's ObjectPAL® in an exemplary embodiment) to a button so that the code executes when the button is clicked." Appellant notes that the program code attached to the button is not graphical source code which comprises a plurality of interconnected nodes, but is text-based source code. For example, FIGs. 4F-4H illustrate an example of the ObjectPAL® text-based source code attached to the button, i.e., the "pushButton" method which executes in response to the button being clicked.

Furthermore, Cain also does not teach, "associating a first user interface event with the first node in response to user input explicitly specifying the first user interface event". As discussed above with reference to claim 1, the graphical user interfaces to which the Examiner refers allow the user to change the built-in methods attached to the button or attach new custom methods to the button, but Cain teaches nothing about associating an event with the button in response to user input explicitly specifying the first event.

For at least the reasons given above, Appellant respectfully submits that Cain does not teach numerous limitations recited in claim 36, and thus, claim 36 is patentably distinct over Cain. Inasmuch as claims 53, 66, and 86 recite similar limitations as claim

36, Appellant respectfully submits that claims 53, 66, and 86 are also patentably distinct over Cain.

Claims 38, 55

Claims 38 and 55 recite additional limitations not taught by Cain. For example, claim 38 recites the additional limitation of, "wherein the plurality of interconnected nodes of the first portion of graphical source code associated with the first node visually indicates functionality for responding to the first user interface event associated with the first node."

As discussed above, Cain teaches that a method comprising text-based source code is attached to the button and executes in response to the button being clicked. Cain does not teach that a first portion of graphical source code comprising a plurality of interconnected nodes is associated with the button and is configured to execute in response to the button being clicked during execution of the program. Cain also does not teach that the plurality of interconnected nodes of the first portion of graphical source code associated with the button visually indicates functionality for responding to the button being clicked. Instead, it is the text-based source code of the method attached to the button that defines the functionality for responding to the button being clicked. Thus, the functionality for responding to the button being clicked is not visually indicated by a plurality of interconnected nodes, but is instead textually indicated by lines of text-based source code.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claims 39, 56

Claims 39 and 56 recite additional limitations not taught by Cain. For example, claim 39 recites the additional limitations of:

> 39. (Previously Presented) The method of claim 36, further comprising:
> creating the first portion of graphical source code in response to user input, wherein creating the first portion of graphical source code comprises:
> displaying the plurality of nodes in response to user input; and
> interconnecting the plurality of nodes in response to user input.

As discussed above, Cain does not teach that a first portion of graphical source code comprising a plurality of interconnected nodes is associated with the first node and is configured to execute in response to the first user interface event associated with the first node being generated during execution of the graphical program. Thus, Cain also does not teach creating the first portion of graphical source code by displaying and interconnecting the nodes in the portion of graphical source code in response to user input.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claims 40, 57

Claims 40 and 57 recite additional limitations not taught by Cain. For example, claim 40 recites the additional limitations of,

> "wherein said associating the first portion of graphical source code with the first node in response to user input comprises displaying the first portion of graphical source code within the first node in response to user input, wherein the first portion of graphical source code associated with the first node is visible in the block diagram of the graphical program."

The Examiner has equated the "first node" with the button taught in Cain. However, Cain does not teach displaying a first portion of graphical source code within the button and does not teach that the first portion of graphical source code is visible in the block diagram of the graphical program. Instead, Cain teaches that the method which executes in response to the button being clicked comprises text-based source code, where the text-based source code is displayed in its own window separately from the block diagram (see FIGs. 4F – 4H).

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claims 41, 44, 58, 60, 67, 78, 83, 84

Claims 41, 44, 58, 60, 67, 78, 83, and 84 recite additional limitations not taught by Cain. For example, claim 41 recites the additional limitations of:

"displaying information explicitly identifying a plurality of user interface events;

wherein said associating the first user interface event with the first node in response to user input explicitly specifying the first user interface event comprises receiving user input selecting the first user interface event from the displayed information."

In the rejection of claim 41 the Examiner asserts that, "users can select and custom create user interface events from the displayed events shown in window 461 of Figure 4E". However, as discussed above with respect to claim 1, the window of FIG. 4E includes a Built-in Methods listbox which lists the built-in (default) methods already attached to the button (Col. 11, lines 62-67) and a Custom Methods box which lists the names of additional custom (user-defined) methods that have been attached to the button (Col. 12, lines 7-9). Cain teaches that, "The behavior of the button object can be varied in two ways: the built-in methods can be edited or new custom methods can be written" (Col. 12, lines 14-16). Thus, the graphical user interface in FIG. 4E allows the user to change the built-in methods attached to the button or attach new custom methods to the button, but Cain teaches nothing about displaying a list of user interface events and associating a first user interface event with the button in response to receiving user input selecting the first user interface event from the displayed list of events.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claims 43, 68, 85

Claims 43, 68, and 85 recite additional limitations not taught by Cain. For example, claim 43 recites the additional limitation of, "wherein the user input explicitly specifying the first user interface event comprises user input explicitly specifying a name of the first user interface event." Cain nowhere teaches the concept of receiving user input that explicitly specifies a name of a first user interface event to associate with the button. The Exaniner again refers to the graphical user interface of FIG. 4E, which, as discussed above, allows the user to change the built-in methods attached to the button or attach new custom methods to the button, but does not allow the user to explicitly specify a name of a first user interface event to associate with the button.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claims 49, 50

Claims 49 and 50 recite additional limitations not taught by Cain. For example, claim 49 recites the additional limitations of:

> "executing the graphical program;
> generating the first user interface event during execution of the graphical program; and
> executing the first portion of graphical source code associated with the first node in response to said generating the first user interface event."

As discussed above, Cain teaches that methods execute in response to events and teaches that the methods are constructed as text-based source code, not graphical source code. Cain does not teach the concept of executing a portion of graphical source code in response to generating a user interface event.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claim 87

Claim 87 recites as follows:

> 87. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:
> creating a first portion of graphical code in response to user input, wherein the first portion of graphical code comprises one or more nodes that visually indicate functionality for responding to programmatic events generated during execution of the graphical program; and
> configuring the graphical program to dynamically register a first programmatic event during execution of the graphical program, wherein dynamically registering the first programmatic event comprises dynamically associating the first programmatic event with the first portion of graphical code;
> wherein said dynamically registering the first programmatic event causes the first portion of graphical code to execute in response to the first programmatic event being generated.

Thus, the method of claim 87 comprises dynamically registering the first programmatic event during execution of the graphical program. Dynamically registering

the first programmatic event causes the first portion of graphical code to execute in response to the first programmatic event being generated.

Cain does not teach dynamically registering the first programmatic event during execution of the graphical program. In the rejection of claim 87, the Examiner refers to the user interacting with the graphical user interface of FIG. 4E. As discussed above, the user interacts with this graphical user interface to change the built-in methods attached to the button or attach new custom methods to the button. The user interacts with the graphical user interface while editing the graphical program, not during execution of the graphical program. Cain teaches nothing at all about an event being dynamically registered during execution of the program.

Furthermore, as discussed above, Cain teaches executing text-based code in response to events being generated, such as the pushButton method shown in FIGS. 4F – 4H. Cain does not teach executing a portion of graphical code in response to an event being generated, where the graphical code comprises one or more nodes that visually indicate functionality for responding to the event. For at least these reasons, Appellant respectfully submits that claim 87 is patentably distinct over Cain.

Claim 88

Claim 88 recites the additional limitations of:

> receiving user input specifying the first programmatic event to be dynamically registered during execution of the graphical program;
>
> wherein the graphical program is configured to dynamically register the first programmatic event during execution of the graphical program in response to the user input specifying the first programmatic event.

The Examiner again refers to the graphical user interface of FIG. 4E. However, as discussed above, this graphical user interface does not allow users to specify an event to be dynamically registered during execution of the graphical program.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Claim 89

Claim 89 recites the additional limitation of, "wherein said configuring the graphical program to dynamically register the first programmatic event during execution of the graphical program comprises configuring a node in the graphical program to dynamically register the first programmatic event during execution of the graphical program." As discussed above, Cain does not teach the concept of dynamically registering an event during execution of a graphical program. Cain also does not teach that the event is dynamically registered by a node in the graphical program.

Appellant thus submits that Cain does not teach the limitations recited in these claims.

Section 103 Rejections

Claims 8 and 9 were rejected under 35 U.S.C. 103(a) as being unpatentable over Cain and U.S. Patent No. 6,578,174 to Zizzo (hereinafter Zizzo). Applicant respectfully traverses these rejections.

Claim 8 recites the limitation of, "wherein, during execution of the graphical program, the block diagram executes on a first computer system and the graphical user interface is displayed on a display of a second computer system." The Examiner asserts that this is taught by Zizzo. Appellant respectfully disagrees. Zizzo does not teach the concept of executing a graphical program, wherein during execution of the graphical program, a block diagram of the graphical program executes on a first computer system and a graphical user interface of the graphical program is displayed on a display of a second computer system.

Claim 9 recites the limitation of, "wherein, during execution of the graphical program, the graphical user interface is displayed on a display of a computer system and the block diagram executes on a reconfigurable instrument connected to the computer system." Zizzo also does not teach the concept of executing a graphical program, wherein during execution of the graphical program, a graphical user interface of the graphical program is displayed on a display of a computer system and a block diagram of the graphical program executes on a reconfigurable instrument connected to the computer system.

Appellant thus submits that the cited references, taken either singly or in combination, do not teach the limitations recited in claims 8 and 9.

Section 112 Rejections

Claims 16, 71, 80-81, and 87 stand rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. Applicant respectfully traverses this rejection.

The Examiner rejected claims 16, 71, and 87 because they previously included negative claim limitations. Appellant submits that these limitations were well supported by the specification, as argued in detail in the response to the Office Action of January 6, 2006. Nevertheless, in the amendment filed on April 11, 2006 together with the Notice of Appeal, Appellant amended claims 16, 71, and 87 to remove their negative limitations. Appellant thus respectfully submits that the Section 112 rejections of claims 16, 71, and 87 are now moot. Appellant believes that the amended claims still adequately express their subject matter without the negative limitations.

Claims 80 and 81 have not been amended to remove negative limitations, and thus the Section 112 rejections remain an issue for these claims. However, Appellant respectfully submits that the Section 112 rejections are erroneous.

Appellant first notes that the Examiner has incorrectly characterized what the MPEP states regarding negative claim limitations. In the Advisory Action of March 14, 2006, the Examiner asserts that, "Negative limitations that exclude limitations must be positively and explicitly recited in the specification (MPEP 2173.05 (i))" (emphasis added). However, what section 2173.05 (i) actually states is that, "Any negative limitation or exclusionary proviso must have basis in the original disclosure." Thus, the MPEP states no requirement that negative limitations must be explicitly recited, but instead says that they must have basis in the original disclosure. Furthermore, the MPEP also states, "Note that a lack of literal basis in the specification for a negative limitation may not be sufficient to establish a prima facie case for lack of descriptive support."

Applicant submits that the specification provides clear basis for the limitations of claims 80 and 81. Claim 80 recites as follows:

80. (Previously Presented) The method of claim 1,
wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node does not include receiving user input specifying a connection between the first node and a second node.

The specification clearly describes an embodiment in which receiving the third user input explicitly specifying the one or more user interface events to configure for the first node does not include receiving user input specifying a connection between the first node and a second node. For example, Figure 11 illustrates an exemplary event configuration dialog. As described on p. 28, line 26 – p. 30, line 12, in one embodiment the user may interact with such an event configuration dialog to configure a set of events for an event structure node. Thus, it is clear to those skilled in the art that in one embodiment, the user interface events to configure for the first node may be specified by user input to a dialog box or other graphical user interface, without requiring the user to specify a connection between the first node and a second node. Appellant thus respectfully submits that the specification provides basis for the limitation recited in claim 80.

Claim 81 recites as follows:

81. (Previously Presented) The method of claim 1,
wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node is performed independently of configuring other nodes in the block diagram of the graphical program.

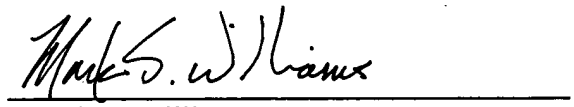
Similarly as discussed above with respect to claim 80, the specification clearly describes an embodiment in which the user may interact with an event configuration dialog to configure a set of events for an event structure node, where the events are configured for the event structure node using the event configuration dialog, independently of configuring other nodes in the block diagram of the graphical program. Appellant thus respectfully submits that the specification provides basis for the limitation recited in claim 81.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-25, 32-44, 46-60, and 62-89 was erroneous, and reversal of the Examiner's decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5150-58700/JCH. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,



Mark S. Williams
Reg. No. 50, 658
Agent for Applicant

Meyertons Hood Kivlin Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: 6-12-06 JCH/JLB

IX. CLAIMS APPENDIX

The following lists the claims as incorporating entered amendments.

1. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:

creating a graphical user interface for the graphical program in response to first user input;

displaying a first node for receiving user interface events in a block diagram for the graphical program in response to second user input;

receiving third user input explicitly specifying one or more user interface events to configure for the first node;

configuring the first node to receive the one or more user interface events explicitly specified by the third user input during execution of the graphical program; and

associating one or more portions of graphical code with the first node in response to fourth user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the first node is configured to receive.

2. (Previously Presented) The method of claim 1,

wherein the first node comprises one or more sub-diagrams, wherein said associating the one or more portions of graphical code with the first node comprises displaying each portion of graphical code within one of the sub-diagrams of the first node.

3. (Previously Presented) The method of claim 2,

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying one or more user interface events to which each of the sub-diagrams of the first node corresponds;

wherein for each portion of graphical code, displaying the portion of graphical code within one of the sub-diagrams of the first node comprises configuring the portion of graphical code to execute in response to the one or more user interface events to which the sub-diagram corresponds.

4. (Previously Presented) The method of claim 3,

wherein for each portion of graphical code, displaying the portion of graphical code within one of the sub-diagrams of the first node comprises displaying the one or more nodes of the portion of graphical code within one of the sub-diagrams of the first node.

5. (Original) The method of claim 1,

wherein the block diagram comprises a data flow block diagram.

6. (Previously Presented) The method of claim 1, further comprising:

executing the graphical program;

wherein one or more user interface events which the first node is configured to receive are generated during execution of the graphical program;

wherein the method further comprises executing one of the portions of graphical code associated with the first node in response to each of the one or more user interface events which the first node is configured to receive being generated during execution of the graphical program.

7. (Original) The method of claim 6,

wherein the one or more user interface events generated during execution of the graphical program are generated in response to user input to the graphical user interface of the graphical program.

8. (Original) The method of claim 6,

wherein, during execution of the graphical program, the block diagram executes on a first computer system and the graphical user interface is displayed on a display of a second computer system.

9. (Original) The method of claim 6,

wherein, during execution of the graphical program, the graphical user interface is displayed on a display of a computer system and the block diagram executes on a reconfigurable instrument connected to the computer system.

10. (Previously Presented) The method of claim 1,

wherein said configuring the first node to receive the one or more user interface events comprises configuring the first node to receive notification when the one or more user interface events are generated during execution of the graphical program.

11. (Previously Presented) The method of claim 1,

wherein said configuring the first node to receive the one or more user interface events comprises configuring the first node to receive information specifying occurrences of the one or more user interface events during execution of the graphical program.

12. (Previously Presented) The method of claim 1,

wherein each of the portions of graphical code associated with the first node is displayed within the first node.

13. (Previously Presented) The method of claim 1,

wherein said configuring the first node to receive the one or more user interface events comprises configuring the first node to receive a first user interface event;

wherein the first user interface event explicitly specifies a first user interface element of the graphical user interface and an action performed on the first user interface element.

14. (Original) The method of claim 13, wherein the first user interface element comprises one of:

an indicator;

a control;

a menu element;

a window.

15. (Previously Presented) The method of claim 1, further comprising:

displaying a first graphical user interface dialog for configuring the first node;

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input to the first graphical user interface dialog to explicitly specify the one or more user interface events.

16. (Previously Presented) The method of claim 1, further comprising:

displaying a second node for dynamically registering user interface events in the block diagram in response to user input;

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying a first user interface event to dynamically register during execution of the graphical program;

wherein the method further comprises configuring the second node to dynamically register the first user interface event during execution of the graphical program such that, after registering the event, the first node becomes operable to receive the first user interface event.

17. (Previously Presented) The method of claim 16,

wherein said configuring the second node to dynamically register the first user interface event during execution of the graphical program comprises connecting the second node to the first node in response to user input.

18. (Previously Presented) The method of claim 1,

wherein the one or more user interface events specified by the third user input includes a first user interface event;

wherein the method further comprises displaying a second node for dynamically un-registering user interface events in the block diagram in response to user input;

configuring the second node to dynamically un-register the first user interface event during execution of the graphical program;

wherein, before said dynamically un-registering the first user interface event, the first node is operable to receive the first user interface event;

wherein, after said dynamically un-registering the first user interface event, the first node is not operable to receive the first user interface event.

19. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:

displaying a first node for receiving programmatic events in a block diagram for the graphical program in response to first user input;

receiving second user input explicitly specifying one or more programmatic events to configure for the first node;

configuring the first node to receive the one or more programmatic events explicitly specified by the second user input during execution of the graphical program; and

associating one or more portions of graphical code with the first node in response to third user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the programmatic events which the first node is configured to receive.

20. (Previously Presented) The method of claim 19,

wherein the first node comprises one or more sub-diagrams, wherein said associating the one or more portions of graphical code with the first node comprises displaying each portion of graphical code within one of the sub-diagrams of the first node.

21. (Previously Presented) The method of claim 20,

wherein said receiving the second user input explicitly specifying the one or more programmatic events to configure for the first node comprises receiving user input explicitly specifying one or more programmatic events to which each of the sub-diagrams of the first node corresponds;

wherein for each portion of graphical code, displaying the portion of graphical code within one of the sub-diagrams of the first node comprises configuring the portion of graphical code to execute in response to the one or more user interface events to which the sub-diagram corresponds.

22. (Original) The method of claim 19, wherein the one or more programmatic events comprise one or more of:

a user interface event;

a system event;

a timer event;

an event generated in response to data acquired from a device.

23. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:

creating a graphical user interface for the graphical program in response to user input;

creating a block diagram for the graphical program in response to user input, wherein creating the block diagram comprises creating a first portion of graphical code in response to user input, wherein the first portion of graphical code includes one or more nodes for responding to user interface events;

receiving user input explicitly specifying one or more user interface events to associate with the first portion of graphical code; and

configuring the first portion of graphical code to execute in response to the one or more explicitly specified user interface events being generated during execution of the graphical program.

24. (Previously Presented) The method of claim 23,

wherein the first portion of graphical code includes a plurality of nodes, wherein said creating the first portion of graphical code in response to user input comprises arranging the plurality of nodes on a display and interconnecting the plurality of nodes in response to user input, wherein the plurality of interconnected nodes visually indicates functionalty for responding to the one or more user interface events associated with the first portion of graphical code.

25. (Previously Presented) The method of claim 23, further comprising:

executing the graphical program;

wherein one or more user interface events associated with the first portion of graphical code are generated during execution of the graphical program;

wherein the method further comprises executing the first portion of graphical code in response to the one or more user interface events associated with the first portion of graphical code being generated.

26. - 31. (Canceled)

32. (Previously Presented) A memory medium for creating a graphical program, the memory medium comprising program instructions executable to:

create a graphical user interface for the graphical program in response to first user input;

display a first node for receiving user interface events in a block diagram for the graphical program in response to second user input;

receive third user input explicitly specifying one or more user interface events to configure for the first node;

configure the first node to receive the one or more user interface events explicitly specified by the third user input during execution of the graphical program; and

associate one or more portions of graphical code with the first node in response to fourth user input, wherein each portion of graphical code comprises one or more nodes for responding to one or more of the user interface events which the first node is configured to receive.

33. (Previously Presented) The memory medium of claim 32,

wherein the first node comprises one or more sub-diagrams, wherein said associating the one or more portions of graphical code with the first node comprises displaying each portion of graphical code within one of the sub-diagrams of the first node.

34. (Previously Presented) The memory medium of claim 33,

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying one or more user interface events to which each of the sub-diagrams of the first node corresponds;

wherein for each portion of graphical code, displaying the portion of graphical code within one of the sub-diagrams of the first node comprises configuring the portion of graphical code to execute in response to the one or more user interface events to which the sub-diagram corresponds.

35. (Previously Presented) The memory medium of claim 34,

wherein for each portion of graphical code, displaying the portion of graphical code within one of the sub-diagrams of the first node comprises displaying the one or more nodes of the portion of graphical code within one of the sub-diagrams of the first node.

36. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:

displaying a first node in a block diagram of the graphical program in response to user input;

associating a first portion of graphical source code with the first node in response to user input, wherein the first portion of graphical source code associated with the first node comprises a plurality of interconnected nodes;

associating a first user interface event with the first node in response to user input explicitly specifying the first user interface event; and

configuring the first portion of graphical source code associated with the first node to execute in response to the first user interface event associated with the first node being generated during execution of the graphical program.

37. (Previously Presented) The method of claim 36,

wherein said first portion of graphical source code executing in response to the first user interface event comprises the first portion of graphical source code executing in response to the first user interface event during execution of the graphical program, wherein the first user interface event is generated during execution of the graphical program.

38. (Previously Presented) The method of claim 36,

wherein the plurality of interconnected nodes of the first portion of graphical source code associated with the first node visually indicates functionality for responding to the first user interface event associated with the first node.

39. (Previously Presented) The method of claim 36, further comprising:

creating the first portion of graphical source code in response to user input, wherein creating the first portion of graphical source code comprises:

displaying the plurality of nodes in response to user input; and

interconnecting the plurality of nodes in response to user input.

40. (Previously Presented) The method of claim 36,

wherein said associating the first portion of graphical source code with the first node in response to user input comprises displaying the first portion of graphical source code within the first node in response to user input, wherein the first portion of graphical source code associated with the first node is visible in the block diagram of the graphical program.

41. (Previously Presented) The method of claim 36, further comprising:

displaying information explicitly identifying a plurality of user interface events;

wherein said associating the first user interface event with the first node in response to user input explicitly specifying the first user interface event comprises receiving user input selecting the first user interface event from the displayed information.

42. (Previously Presented) The method of claim 36, further comprising:

associating a second portion of graphical source code with the first node in response to user input;

associating a second user interface event with the first node in response to user input explicitly specifying the second user interface event; and

configuring the second portion of graphical source code associated with the first node to execute in response to the second user interface event associated with the first node.

43. (Previously Presented) The method of claim 36,

wherein the user input explicitly specifying the first user interface event comprises user input explicitly specifying a name of the first user interface event.

44. (Previously Presented) The method of claim 36, further comprising:

displaying a graphical user interface dialog, wherein the graphical user interface dialog explicitly displays a plurality of user interface events;

wherein said receiving user input explicitly specifying the first user interface event comprises receiving user input via the graphical user interface dialog to select the first user interface event from the plurality of explicitly displayed user interface events.

45. (Canceled).

46. (Previously Presented) The method of claim 36,

wherein the graphical program has a graphical user interface;

wherein the first user interface event is associated with a first user interface element of the graphical user interface.

47. (Previously Presented) The method of claim 46, wherein the first user interface element comprises one of:

an indicator;

a control;

a menu element;

a window.

48. (Previously Presented) The method of claim 36,

wherein the graphical program includes a graphical user interface;

wherein the first user interface event is associated with a user action performed on the graphical user interface.

49. (Previously Presented) The method of claim 36, further comprising:

executing the graphical program;

generating the first user interface event during execution of the graphical program; and

executing the first portion of graphical source code associated with the first node in response to said generating the first user interface event.

50. (Previously Presented) The method of claim 36,

wherein the graphical program includes a graphical user interface;

wherein the method further comprises:

executing the graphical program;

generating the first user interface event during execution of the graphical program, wherein said generating the first user interface event comprises generating the first user interface event in response to user input to the graphical user interface; and

executing the first portion of graphical source code associated with the first node in response to said generating the first user interface event.

51. (Previously Presented) The method of claim 36,

wherein the block diagram of the graphical program comprises a data flow diagram.

52. (Previously Presented) The method of claim 36,

wherein the block diagram comprises a plurality of interconnected nodes that visually indicate functionality of the graphical program.

53. (Previously Presented) A memory medium for creating a graphical program, the memory medium comprising program instructions executable to:

display a first node in a block diagram of the graphical program in response to user input;

associate a first portion of graphical source code with the first node in response to user input, wherein the first portion of graphical source code associated with the first node comprises a plurality of interconnected nodes;

associate a first user interface event with the first node in response to user input explicitly specifying the first user interface event; and

configure the first portion of graphical source code associated with the first node to execute in response to the first user interface event associated with the first node being generated during execution of the graphical program.

54. (Previously Presented) The memory medium of claim 53,

wherein said first portion of graphical source code executing in response to the first user interface event comprises the first portion of graphical source code executing in response to the first user interface event during execution of the graphical program, wherein the first user interface event is generated during execution of the graphical program.

55. (Previously Presented) The memory medium of claim 53,

wherein the plurality of interconnected nodes of the first portion of graphical source code associated with the first node visually indicates functionality for responding to the first user interface event associated with the first node.

56. (Previously Presented) The memory medium of claim 53, wherein the program instructions are further executable to:

create the first portion of graphical source code in response to user input, wherein creating the first portion of graphical source code comprises:

displaying the plurality of nodes in response to user input; and

interconnecting the plurality of nodes in response to user input.

57. (Previously Presented) The memory medium of claim 53,

wherein said associating the first portion of graphical source code with the first node in response to user input comprises displaying the first portion of graphical source code within the first node in response to user input, wherein the first portion of graphical source code associated with the first node is visible in the block diagram of the graphical program.

58. (Previously Presented) The memory medium of claim 53, wherein the program instructions are further executable to:

display information explicitly identifying a plurality of user interface events;

wherein said associating the first user interface event with the first node in response to user input explicitly specifying the first user interface event comprises

receiving user input selecting the first user interface event from the displayed information.

59. (Previously Presented) The memory medium of claim 53, wherein the program instructions are further executable to:

associate a second portion of graphical source code with the first node in response to user input;

associate a second user interface event with the first node in response to user input explicitly specifying the second user interface event; and

configure the second portion of graphical source code associated with the first node to execute in response to the second user interface event associated with the first node.

60. (Previously Presented) The memory medium of claim 53,

wherein the memory medium further comprises program instructions executable to display a graphical user interface dialog, wherein the graphical user interface dialog explicitly displays a plurality of user interface events;

wherein said receiving user input explicitly specifying the first user interface event comprises receiving user input via the graphical user interface dialog to select the first user interface event from the plurality of explicitly displayed user interface events.

61. (Canceled)

62. (Previously Presented) The memory medium of claim 53,

wherein the graphical program includes a graphical user interface;

wherein the first user interface event is associated with a first user interface element of the graphical user interface.

63. (Previously Presented) The memory medium of claim 53,

wherein the graphical program includes a graphical user interface;

wherein the first user interface event is associated with a user action performed on the graphical user interface.

64. (Previously Presented) The memory medium of claim 53,

wherein the block diagram of the graphical program comprises a data flow diagram.

65. (Previously Presented) The memory medium of claim 53,

wherein the block diagram comprises a plurality of interconnected nodes that visually indicate functionality of the graphical program.

66. (Previously Presented) A system for creating a graphical program, the system comprising:

a memory storing program instructions;

a processor coupled to the memory; and

a display device;

wherein the processor is operable to execute the program instructions stored in the memory to:

display a first node on the display device in response to user input, wherein said displaying the first node comprises displaying the first node in a block diagram of the graphical program;

associate a first portion of graphical source code with the first node in response to user input, wherein the first portion of graphical source code associated with the first node comprises a plurality of interconnected nodes;

associate a first user interface event with the first node in response to user input explicitly specifying the first user interface event; and

configure the first portion of graphical source code associated with the first node to execute in response to the first user interface event associated with the first node being generated during execution of the graphical program.

67. (Previously Presented) The method of claim 1, further comprising:

displaying a list of user interface events;

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input to select the one or more user interface events from the displayed list of user interface events.

68. (Previously Presented) The method of claim 1,

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input specifying names of the one or more user interface events to configure for the first node.

69. (Previously Presented) The method of claim 1,

wherein said associating the one or more portions of graphical code with the first node comprises associating a first portion of graphical code and a second portion of graphical code with the first node;

wherein the first node comprises a plurality of sub-diagrams;

wherein associating the first portion of graphical code with the first node comprises displaying the first portion of graphical code within a first sub-diagram of the first node; and

wherein associating the second portion of graphical code with the first node comprises displaying the second portion of graphical code within a second sub-diagram of the first node.

70. (Previously Presented) The method of claim 1, further comprising:

displaying one or more of the portions of graphical code associated with the first node in the block diagram of the graphical program.

71. (Previously Presented) The method of claim 1,

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input

explicitly specifying a first user interface event to be dynamically registered during execution of the graphical program;

wherein the method further comprises configuring the graphical program to dynamically register the first user interface event during execution of the graphical program;

wherein said dynamically registering the first user interface event enables the first node to receive the first user interface event.

72. (Previously Presented) The method of claim 1,

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying one or more user interface events to which each of the portions of graphical code associated with the first node corresponds.

73. (Previously Presented) The method of claim 1,

wherein each of the portions of graphical code is comprised within the first node.

74. (Previously Presented) The method of claim 1,

wherein each of the portions of graphical code is located separately from the first node.

75. (Previously Presented) The method of claim 1,

wherein for each of the portions of graphical code, said associating the portion of graphical code with the first node comprises associating the portion of graphical code with one or more of the user interface events which the first node is configured to receive, wherein the portion of graphical code comprises one or more nodes for responding to the one or more user interface events with which the portion of graphical code is associated.

76. (Previously Presented) The method of claim 1,

wherein said receiving third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input explicitly specifying a first user interface event and a second user interface event;

wherein said associating the one or more portions of graphical code with the first node comprises associating a first portion of graphical code with the first node, wherein the first portion of graphical code comprises a plurality of interconected nodes visually indicating functionality for responding to the first user interface event;

wherein said associating the one or more portions of graphical code with the first node further comprises associating a second portion of graphical code with the first node, wherein the second portion of graphical code comprises a plurality of interconected nodes visually indicating functionality for responding to the second user interface event.

77. (Previously Presented) The method of claim 76, further comprising:

executing the graphical program;

executing the first portion of graphical code associated with the first node in response to the first user interface event being generated during execution of the graphical program; and

executing the second portion of graphical code associated with the first node in response to the second user interface event being generated during execution of the graphical program.

78. (Previously Presented) The method of claim 1, further comprising:

displaying information explicitly identifying a plurality of user interface events;

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises receiving user input selecting the one or more user interface events from the displayed information.

79. (Previously Presented) The method of claim 1,

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node comprises graphically connecting a

plurality of objects to the first node in response to user input, wherein each object directly represents a user interface event.

80. (Previously Presented) The method of claim 1,

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node does not include receiving user input specifying a connection between the first node and a second node.

81. (Previously Presented) The method of claim 1,

wherein said receiving the third user input explicitly specifying the one or more user interface events to configure for the first node is performed independently of configuring other nodes in the block diagram of the graphical program.

82. (Previously Presented) The method of claim 1, further comprising:

creating the portions of graphical code in response to user input, wherein for each portion of graphical code, creating the portion of graphical code comprises including one or more nodes in the portion of graphical code in response to user input, wherein the one or more nodes are operable to respond to one or more of the user interface events which the first node is configured to receive.

83. (Previously Presented) The method of claim 23, further comprising:

displaying information explicitly identifying a plurality of user interface events;

wherein said receiving the user input explicitly specifying the one or more user interface events to associate with the first portion of graphical code comprises receiving user input selecting the one or more user interface events from the displayed information.

84. (Previously Presented) The method of claim 23, further comprising:

displaying a list of user interface events;

wherein said receiving the user input explicitly specifying the one or more user interface events to associate with the first portion of graphical code comprises receiving

user input selecting the one or more user interface events from the displayed list of user interface events.

85. (Previously Presented) The method of claim 23,

wherein said receiving the user input explicitly specifying the one or more user interface events to associate with the first portion of graphical code comprises receiving user input specifying names of the one or more user interface events to associate with the first portion of graphical code.

86. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:

displaying a first node for receiving programmatic events in a block diagram of the graphical program in response to user input;

associating a plurality of portions of graphical code with the first node in response to user input, wherein each portion of graphical code comprises one or more nodes;

receiving user input explicitly specifying one or more programmatic events to which each of the portions of graphical code associated with the first node corresponds; and

for each of the portions of graphical code associated with the first node, configuring the portion of graphical code to execute in response to the one or more programmatic events to which the portion of graphical code corresponds being generated during execution of the graphical program.

87. (Previously Presented) A computer-implemented method for creating a graphical program, the method comprising:

creating a first portion of graphical code in response to user input, wherein the first portion of graphical code comprises one or more nodes that visually indicate functionality for responding to programmatic events generated during execution of the graphical program; and

configuring the graphical program to dynamically register a first programmatic event during execution of the graphical program, wherein dynamically registering the

first programmatic event comprises dynamically associating the first programmatic event with the first portion of graphical code;

wherein said dynamically registering the first programmatic event causes the first portion of graphical code to execute in response to the first programmatic event being generated.

88. (Previously Presented) The method of claim 87, further comprising:

receiving user input specifying the first programmatic event to be dynamically registered during execution of the graphical program;

wherein the graphical program is configured to dynamically register the first programmatic event during execution of the graphical program in response to the user input specifying the first programmatic event.

89. (Previously Presented) The method of claim 87,

wherein said configuring the graphical program to dynamically register the first programmatic event during execution of the graphical program comprises configuring a node in the graphical program to dynamically register the first programmatic event during execution of the graphical program.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.